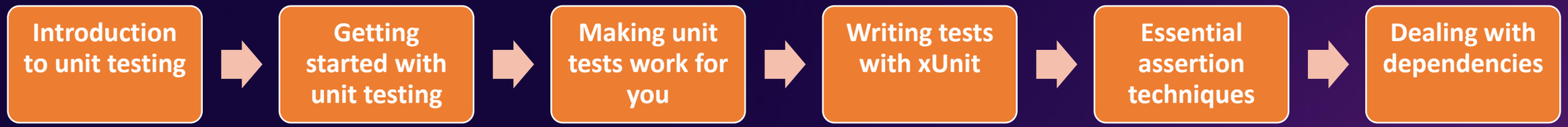
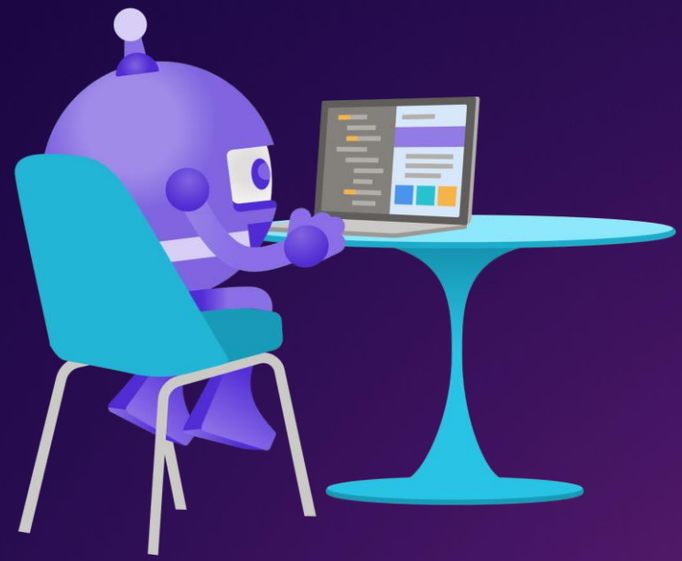
Course Outline



Is this course for you?

Basic C# or Java Knowledge



Beginner Level Course

Software prerequisites

+

.NET SDK
https://dot.net

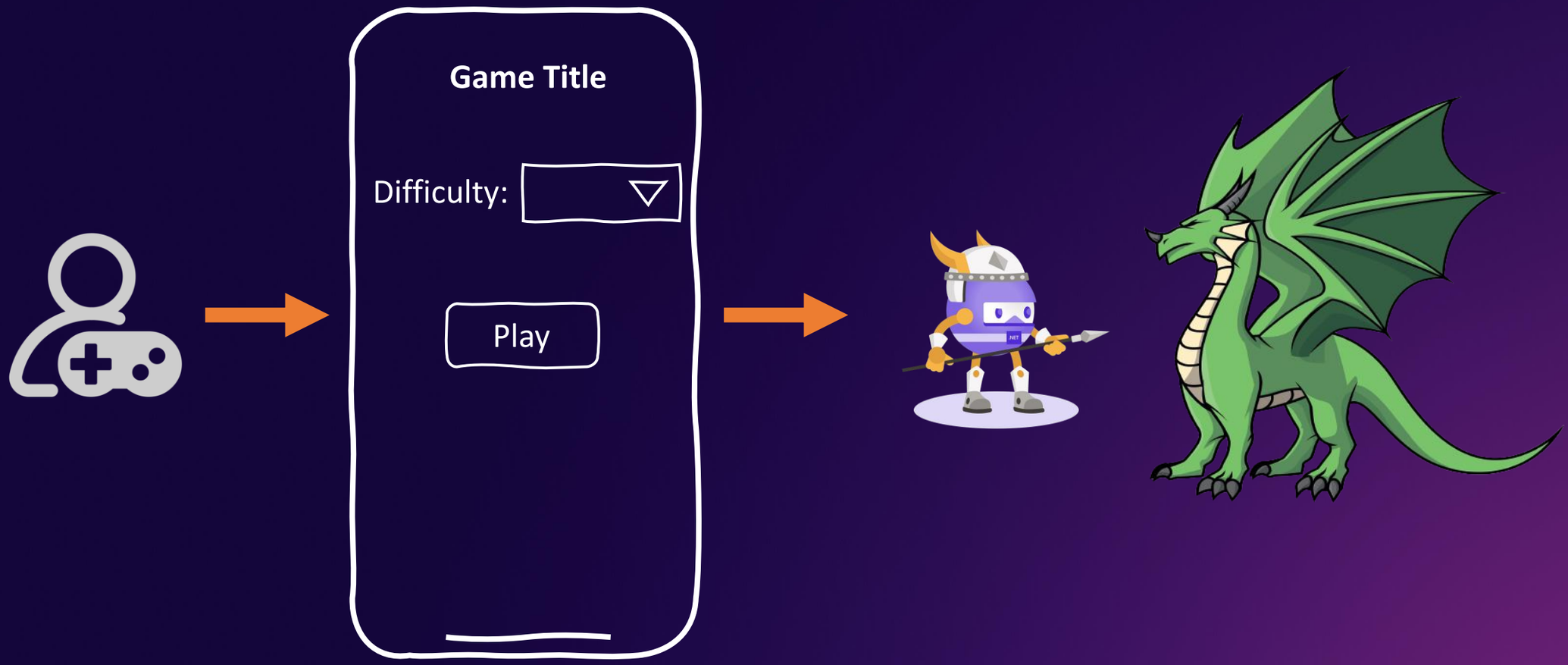
Visual Studio Code
https://code.visualstudio.com

OR

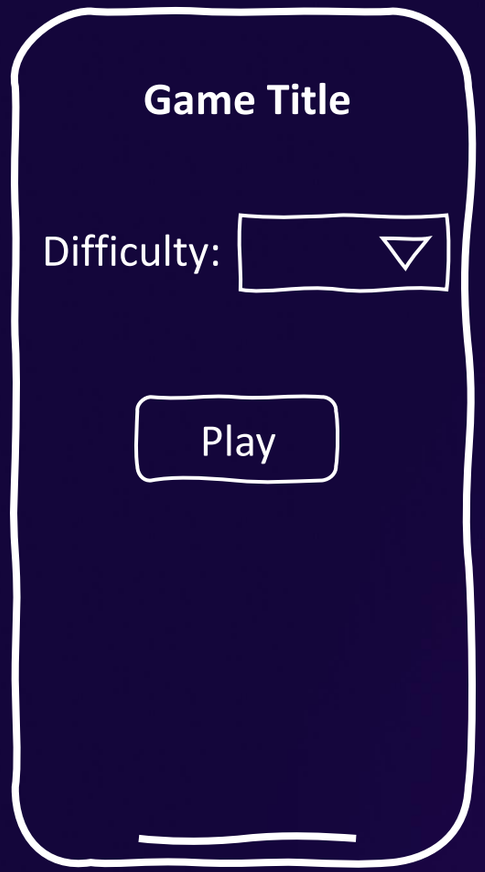
Visual Studio
https://visualstudio.microsoft.com

What is automated testing?

A simple game



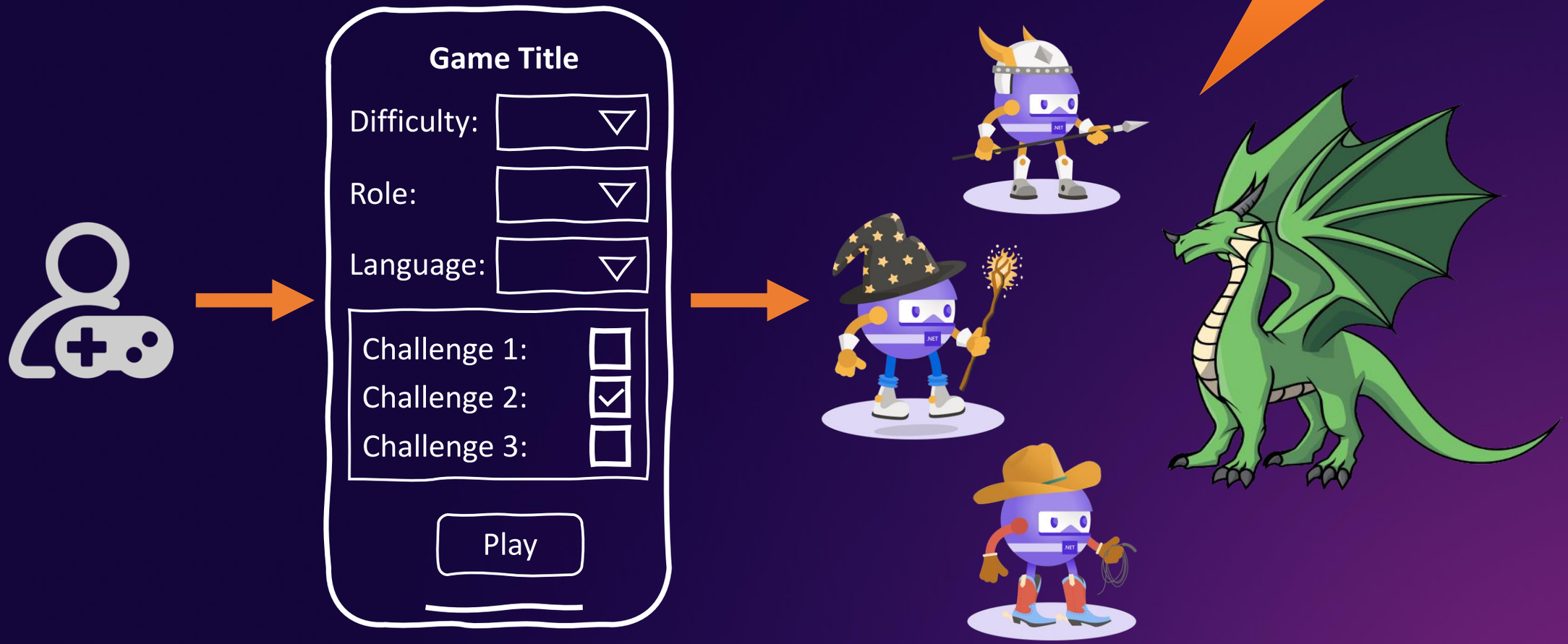
Testing the game UI



Verification Steps

- ✓ Shows expected difficulty levels
- ✓ Defaults to medium difficulty
- ✓ Play button starts game
- ✓ Started game matches difficulty
- ✓ Game starts in less than 2 secs

Game requirements evolve



Manual testing becomes challenging

Verification Steps

- ✓ Shows expected difficulty levels
- ✓ Defaults to medium difficulty
- ✓ Shows expected roles
- ✓ Defaults to warrior role
- ✓ Shows available languages
- ✓ Defaults to English language
- ✓ All available challenges show up
- ✓ Can select multiple challenges
- ✓ Play button starts game
- ✓ Game matches difficulty
- ✓ Game matches one of the selected challenges
- ✓ Player is assigned selected role
- ✓ Game starts in less than 5 secs
- ✓ Players speak selected language

Challenges

- Very time consuming
- Limited coverage of complex scenarios
- Inconsistent results due to human errors
- Need multiple testers to cover various scenarios
- Depends on testers skills
- Slow feedback for devs

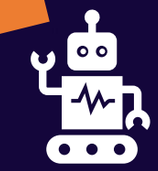
What is Automated Testing?

The use of software tools to execute predefined test cases on an application without human intervention

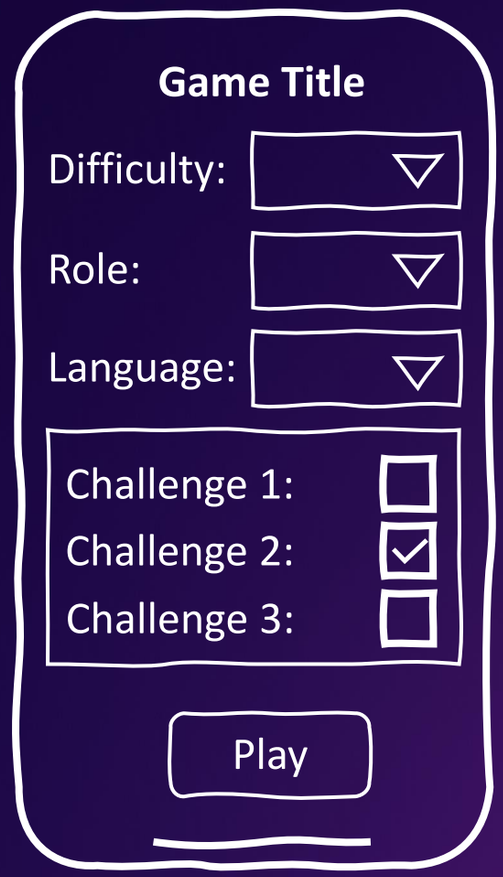
Switching to automated testing

Verification Steps

- ✓ Shows expected difficulty levels
- ✓ Defaults to medium difficulty
- ✓ Shows expected roles
- ✓ Defaults to warrior role
- ✓ Shows available languages
- ✓ Defaults to English language
- ✓ All available challenges show up
- ✓ Can select multiple challenges
- ✓ Play button starts game
- ✓ Game matches difficulty
- ✓ Game matches one of the selected challenges
- ✓ Player is assigned selected role
- ✓ Game starts in less than 5 secs
- ✓ Players speak selected language



Test Program

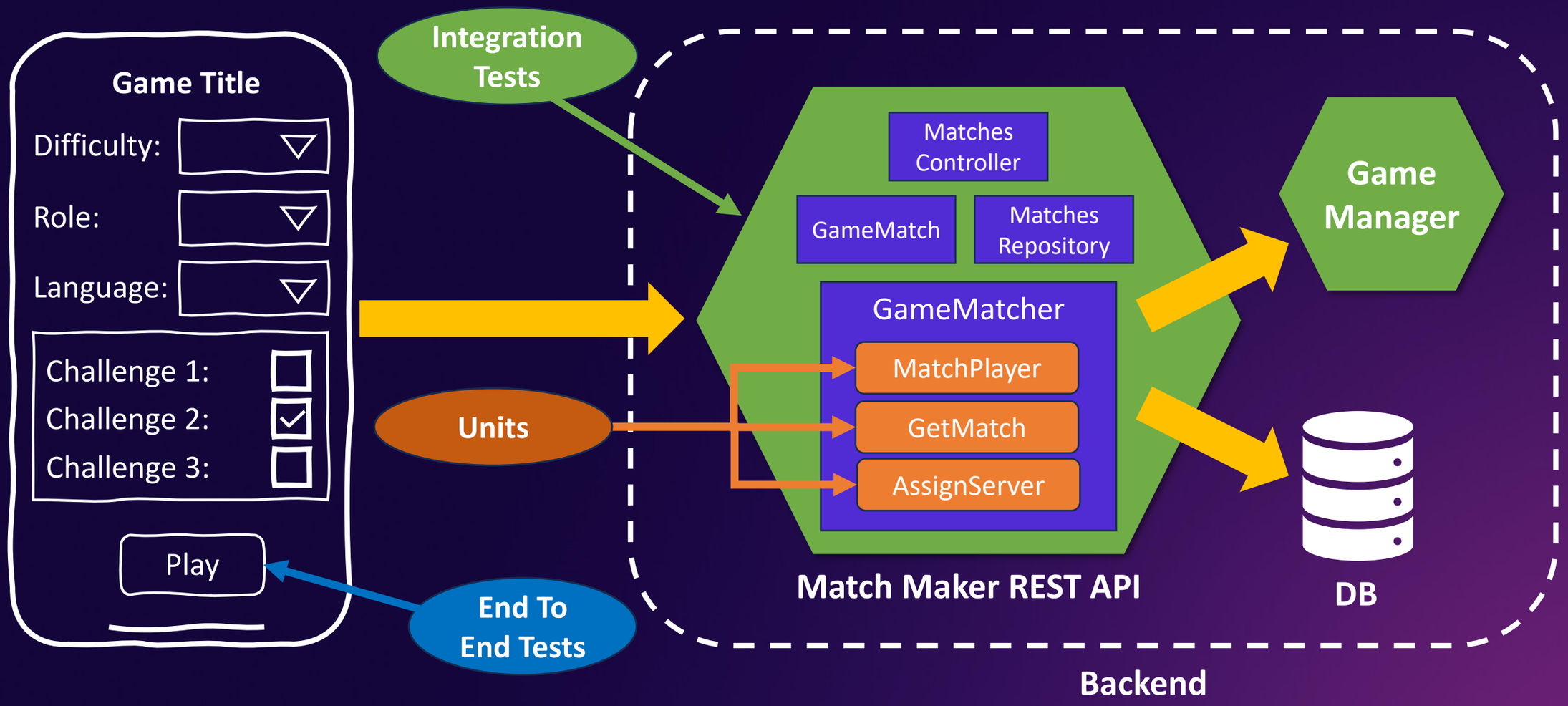


Benefits

- Tests can run fast and frequently
- Extensive coverage of complex scenarios
- Consistent verification steps reduce human errors
- Provides rapid feedback to dev team
- Cost effective
- Can run outside regular hours

Types of automated tests

Many things to test



juliocasal.com

Different types of tests

THIS COURSE

Unit Tests

- ✓ Verify a small piece of code
- ✓ Does it quickly
- ✓ Does it in isolation

Integration Tests

- ✓ Verify the interactions between different units
- ✓ Focus on integration with out-of-process dependencies

End To End Tests

- ✓ Verify the application from start to finish
- ✓ Validate from the user's perspective

Performance Tests

Security Tests

Smoke Tests

Acceptance Tests

Load Tests

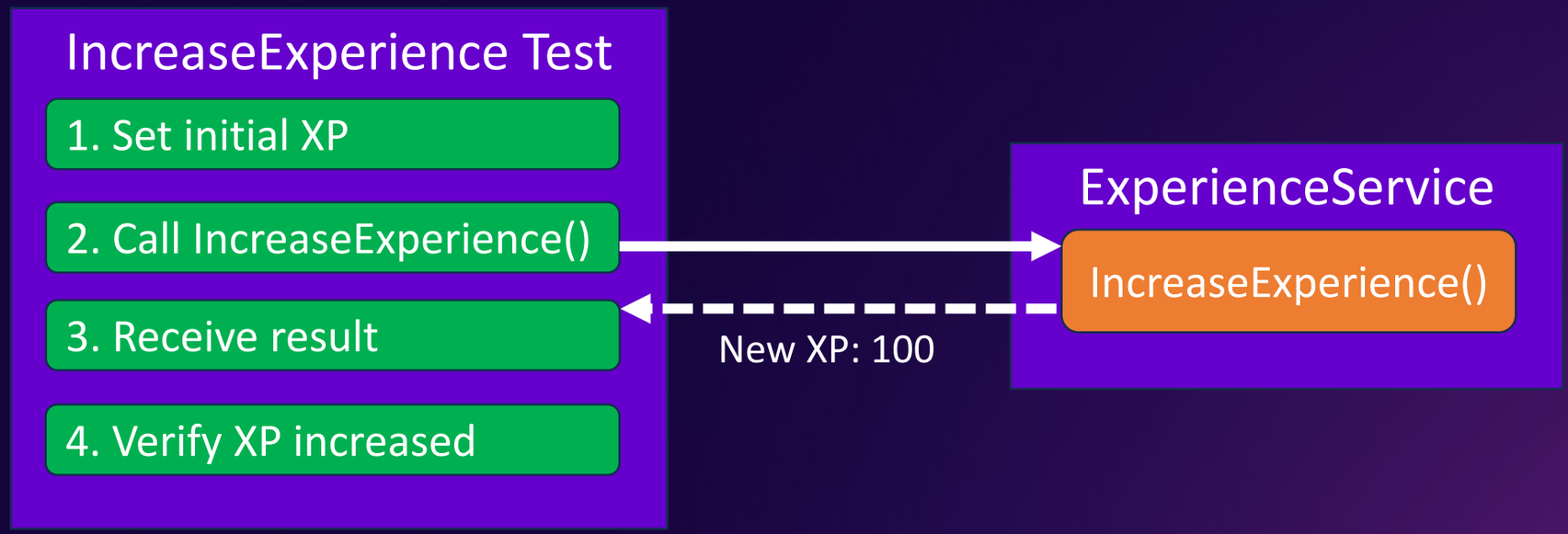
Usability Tests

What is unit testing?

What is Unit Testing?

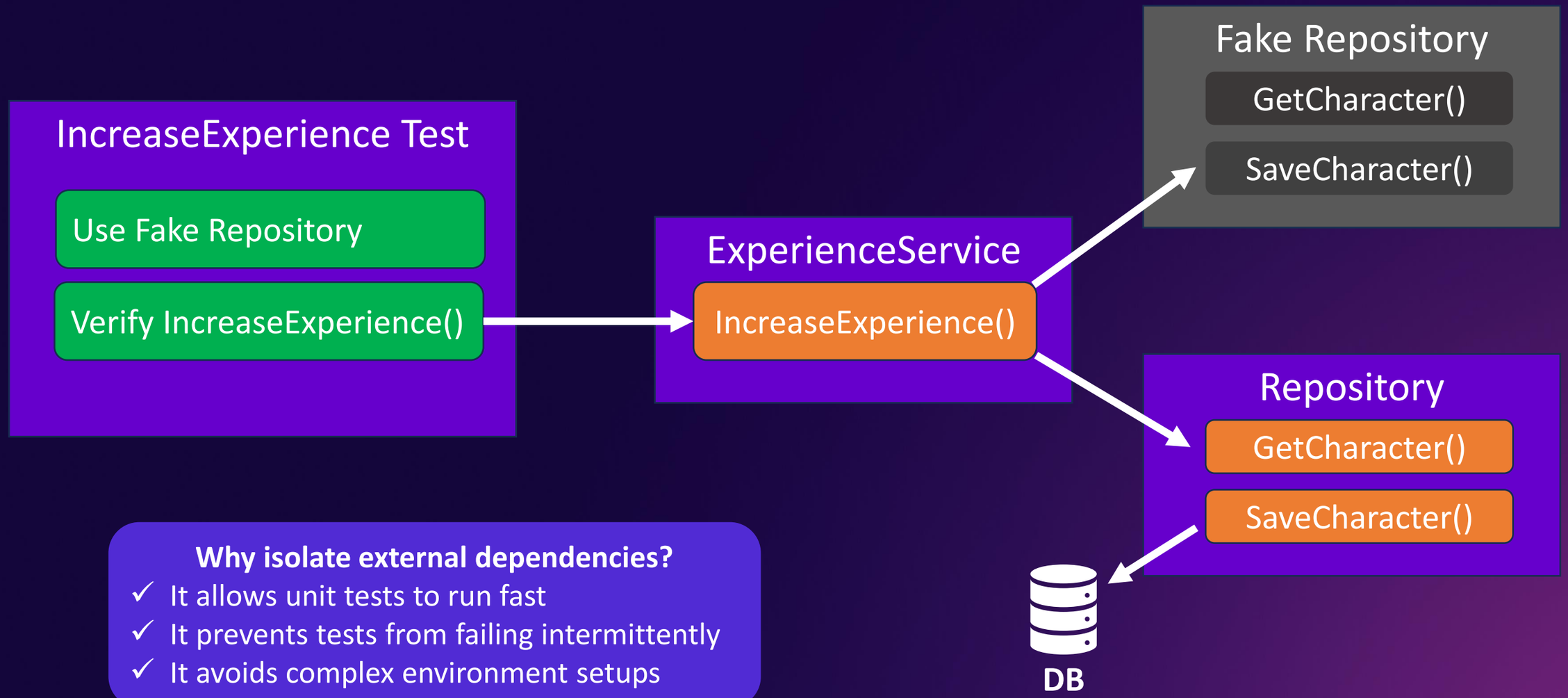
A software testing technique where individual units of an application are tested in isolation to ensure that they function correctly.

Testing units



- ✓ A unit test verifies a unit of the application is working as expected
- ✓ The test usually runs in the order of milliseconds

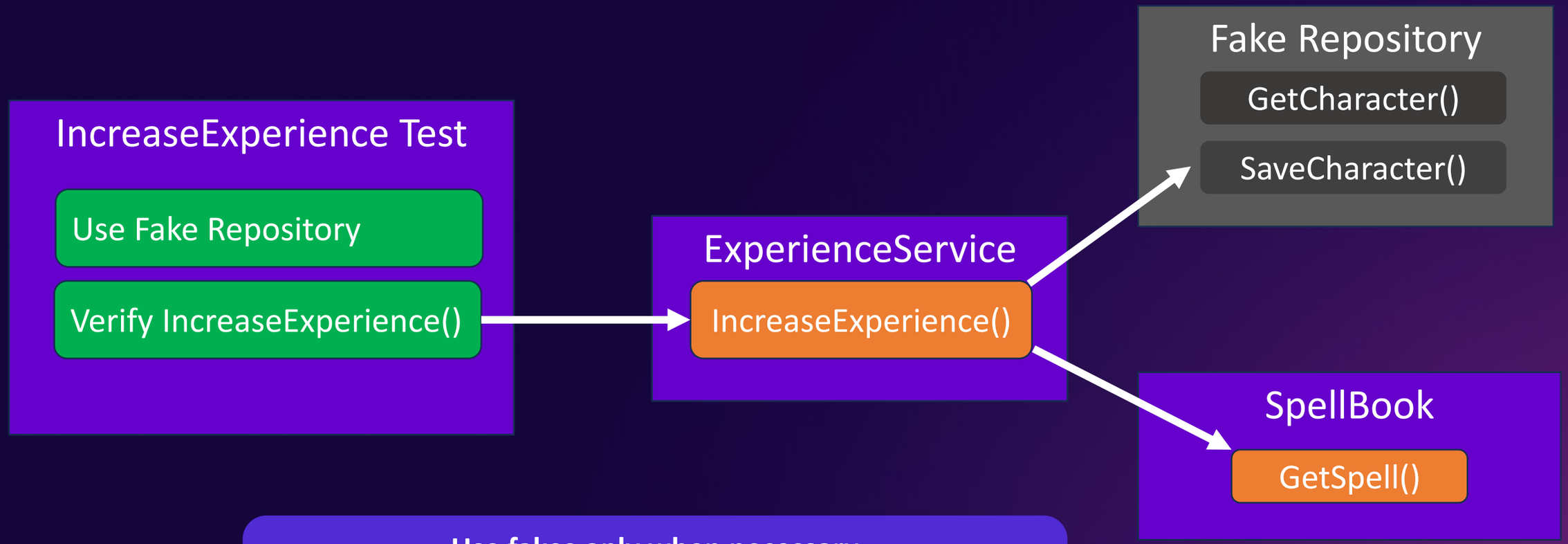
Isolating units from external dependencies



Why isolate external dependencies?

- ✓ It allows unit tests to run fast
- ✓ It prevents tests from failing intermittently
- ✓ It avoids complex environment setups

No need to fake every dependency

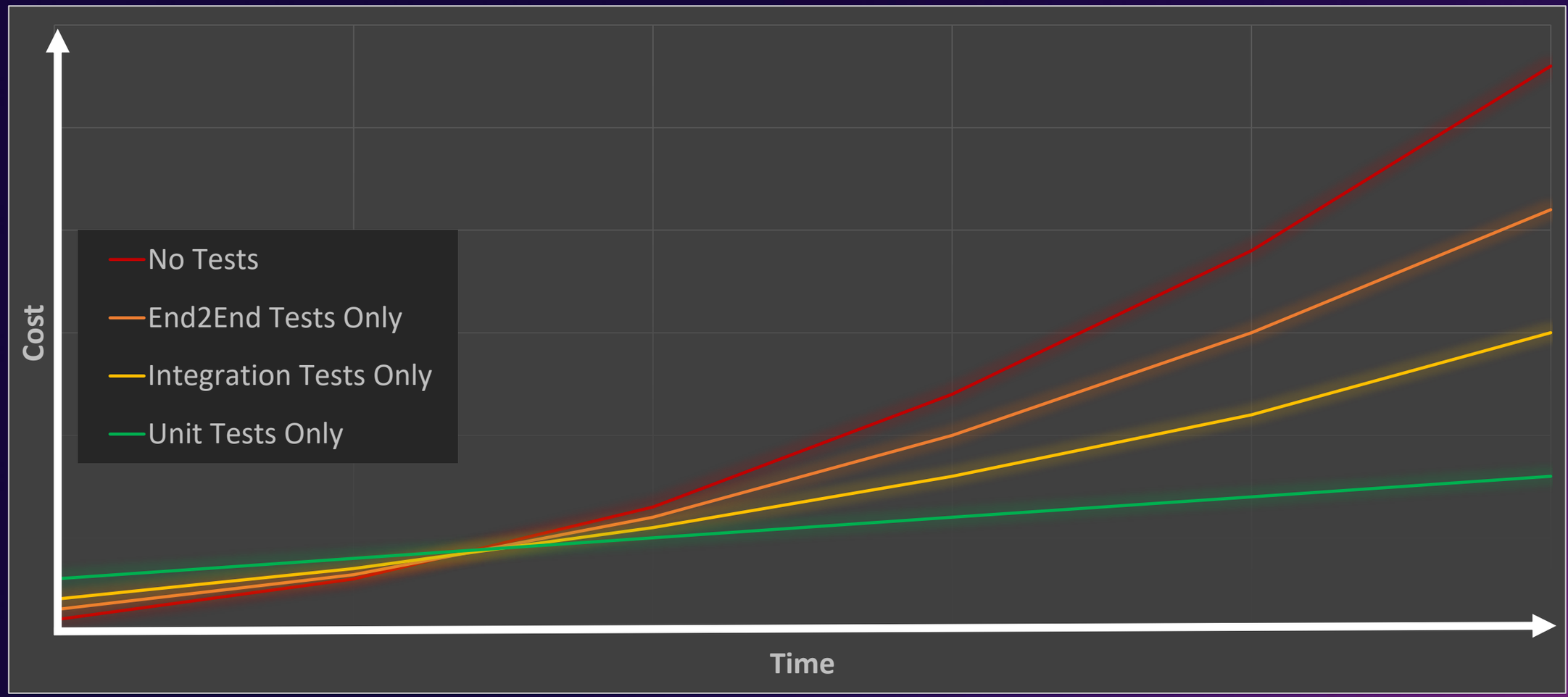


Use fakes only when necessary

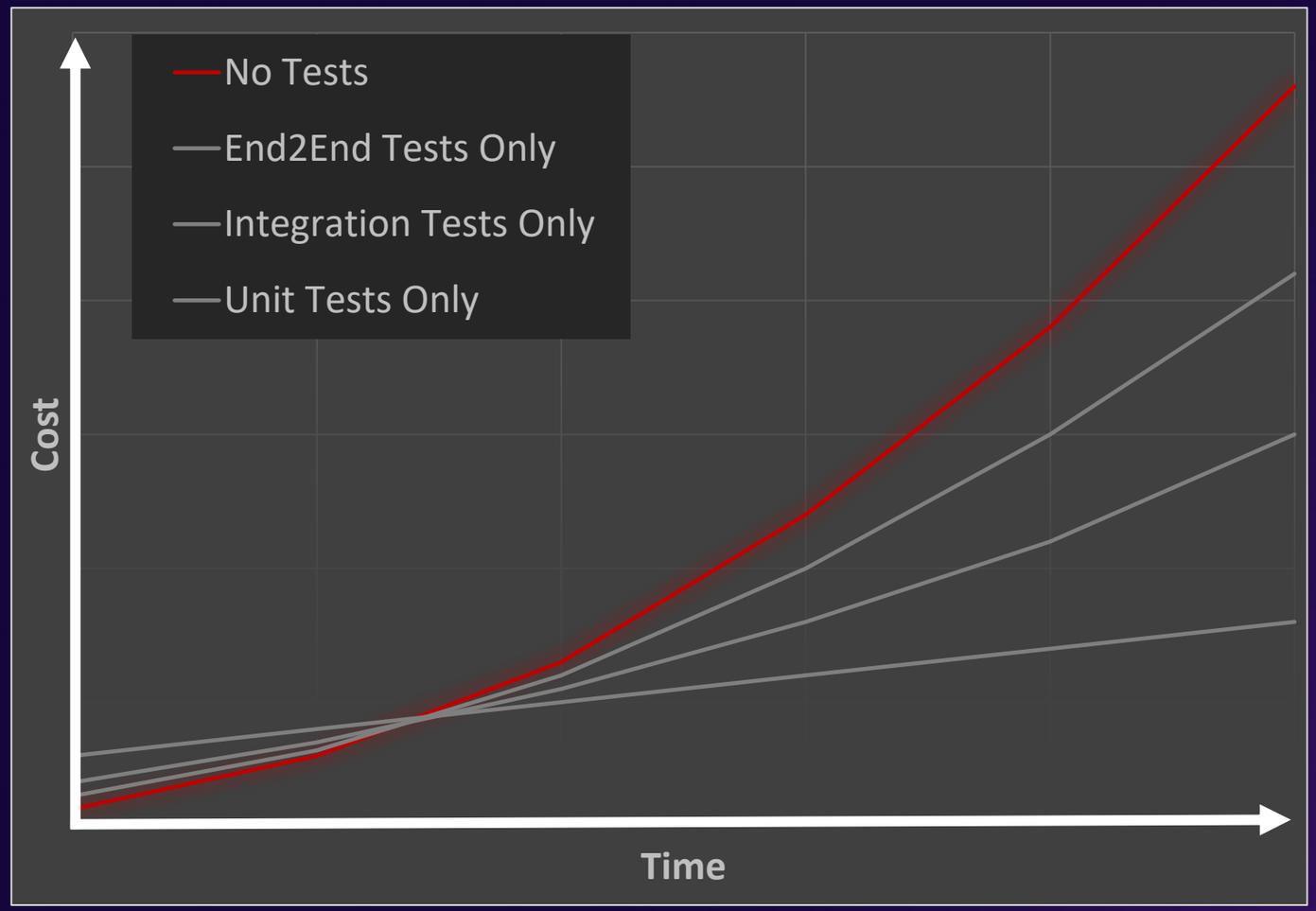
- ✓ It makes tests easier to setup and understand
- ✓ It prevents coupling tests to implementation details

Do I need unit tests?

The relative cost of writing software over time



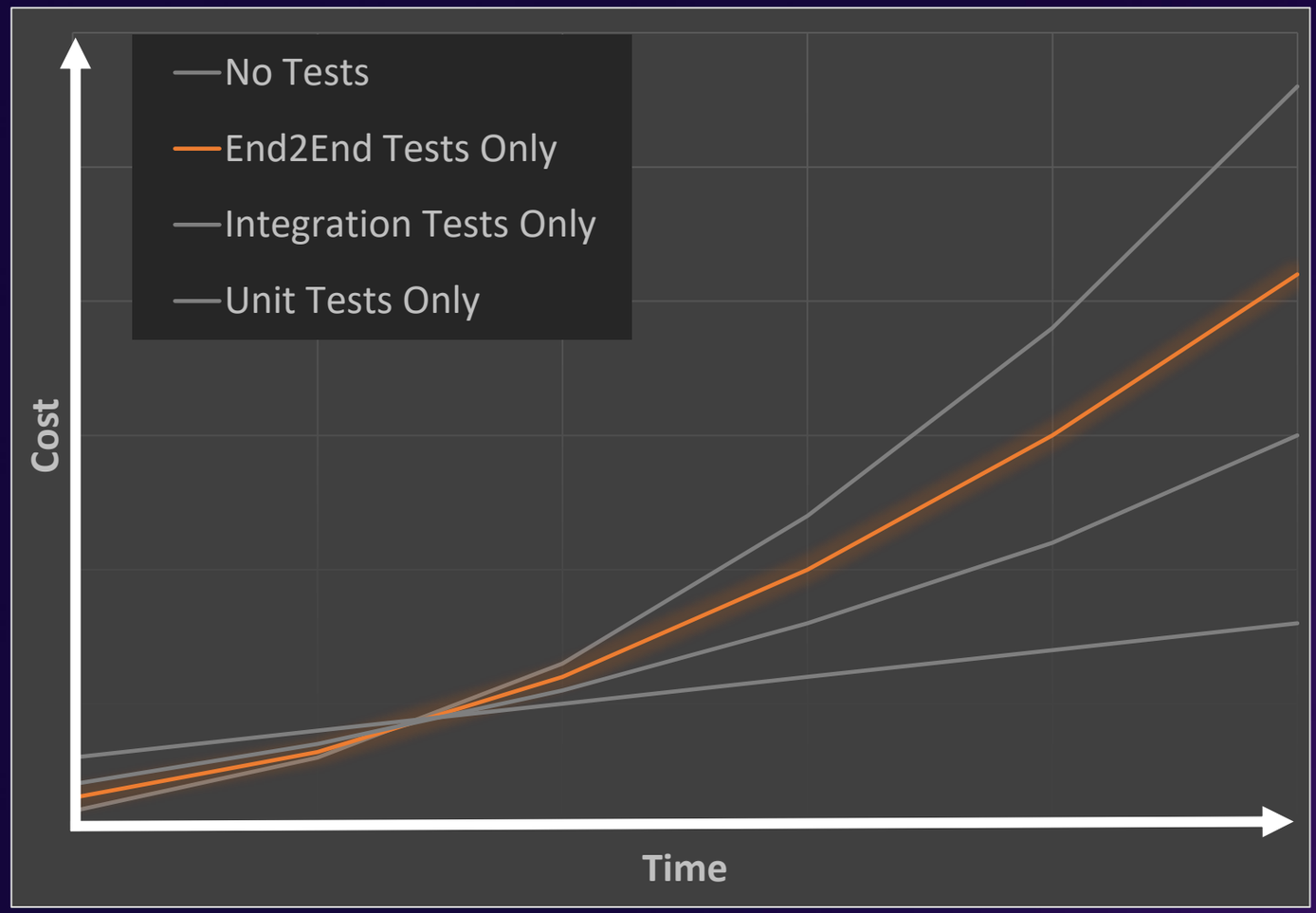
The relative cost of not having any tests



No Tests

- Bugs show up in production
- The company and customers may lose revenue
- Tons of engineer hours spent debugging and bug fixing
- Rolling back is complex, time consuming and can bring more issues
- The company reputation is damaged

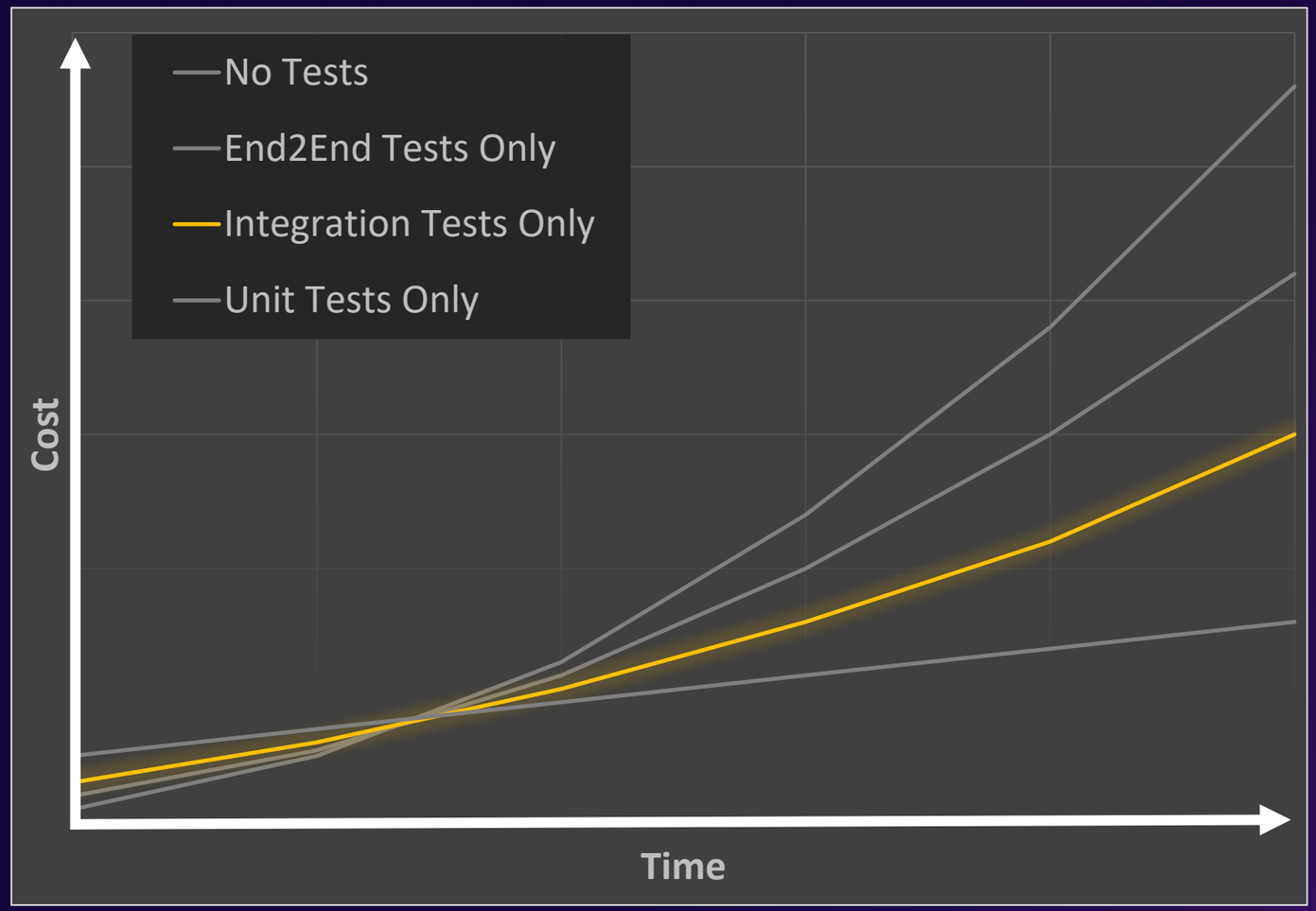
The relative cost of having end to end tests only



End To End Tests Only

- Require standing up a complete (expensive) environment
- Long feedback cycle to find, fix bugs and verify fixes
- Flaky due to UI and external services which randomizes devs
- Hard to diagnose issues, which consumes dev hours
- Hard to maintain tests take lots of dev effort

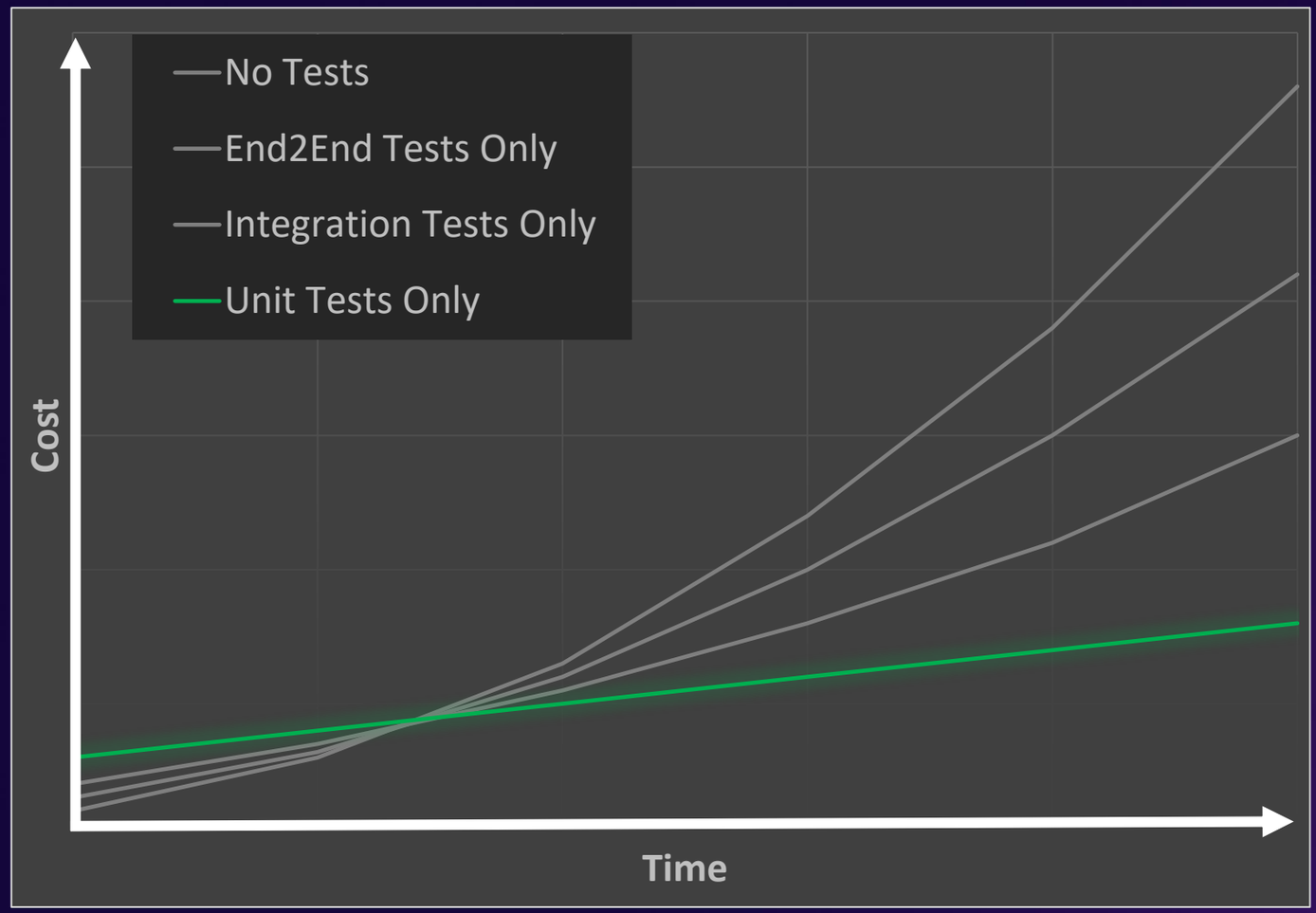
The relative cost of having integration tests only



Integration Tests Only

- Faster feedback cycle than E2E tests, but still slow since mostly run con CI/CD pipelines
- More stable than E2E tests, but external dependencies can still randomize devs
- Easier to diagnose issues than E2E tests, but still can’t spot the exact issue easily
- Easier to maintain than E2E tests, but can still require complex setup

The relative cost of having unit tests only

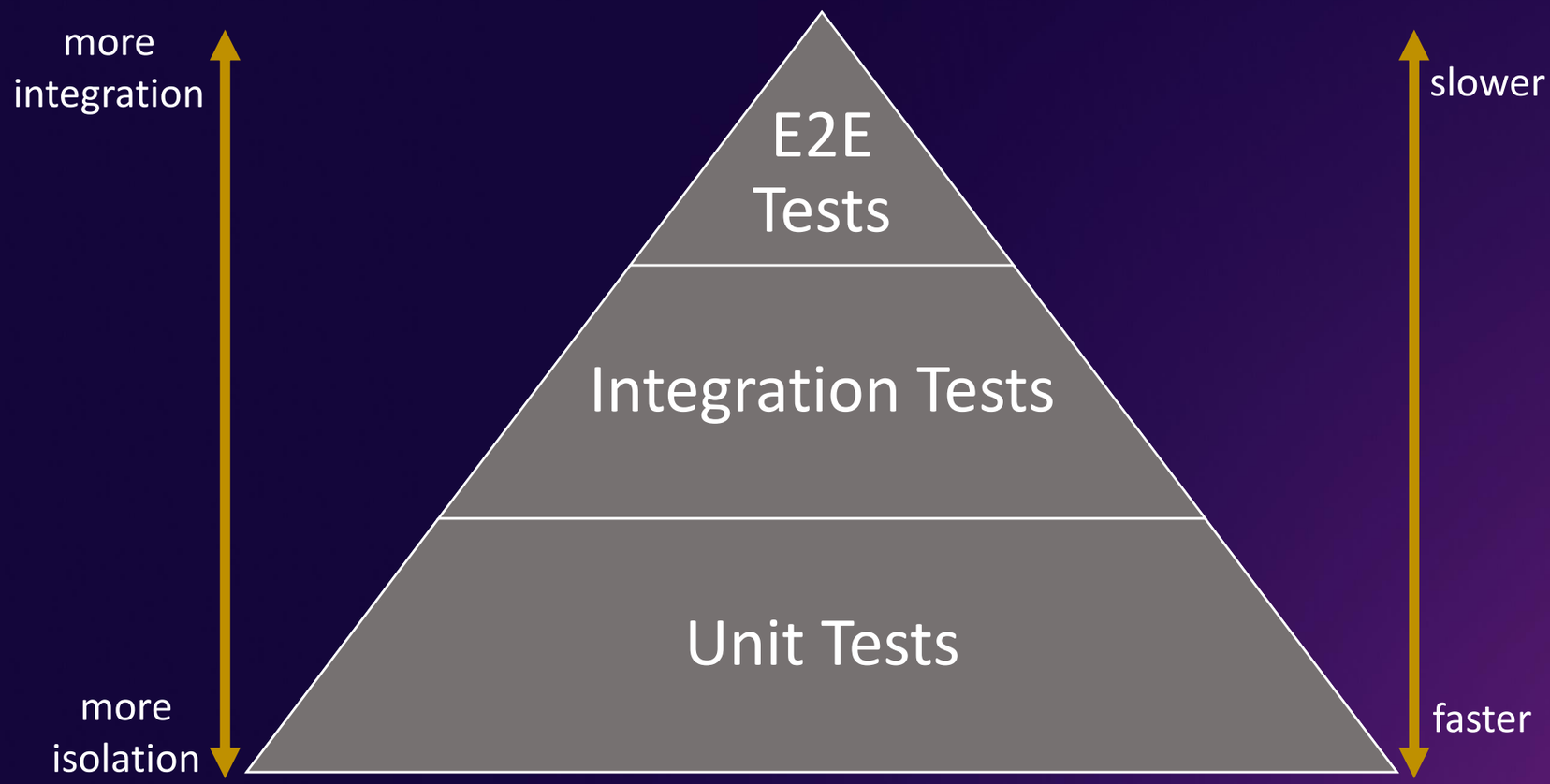


Unit Tests Only

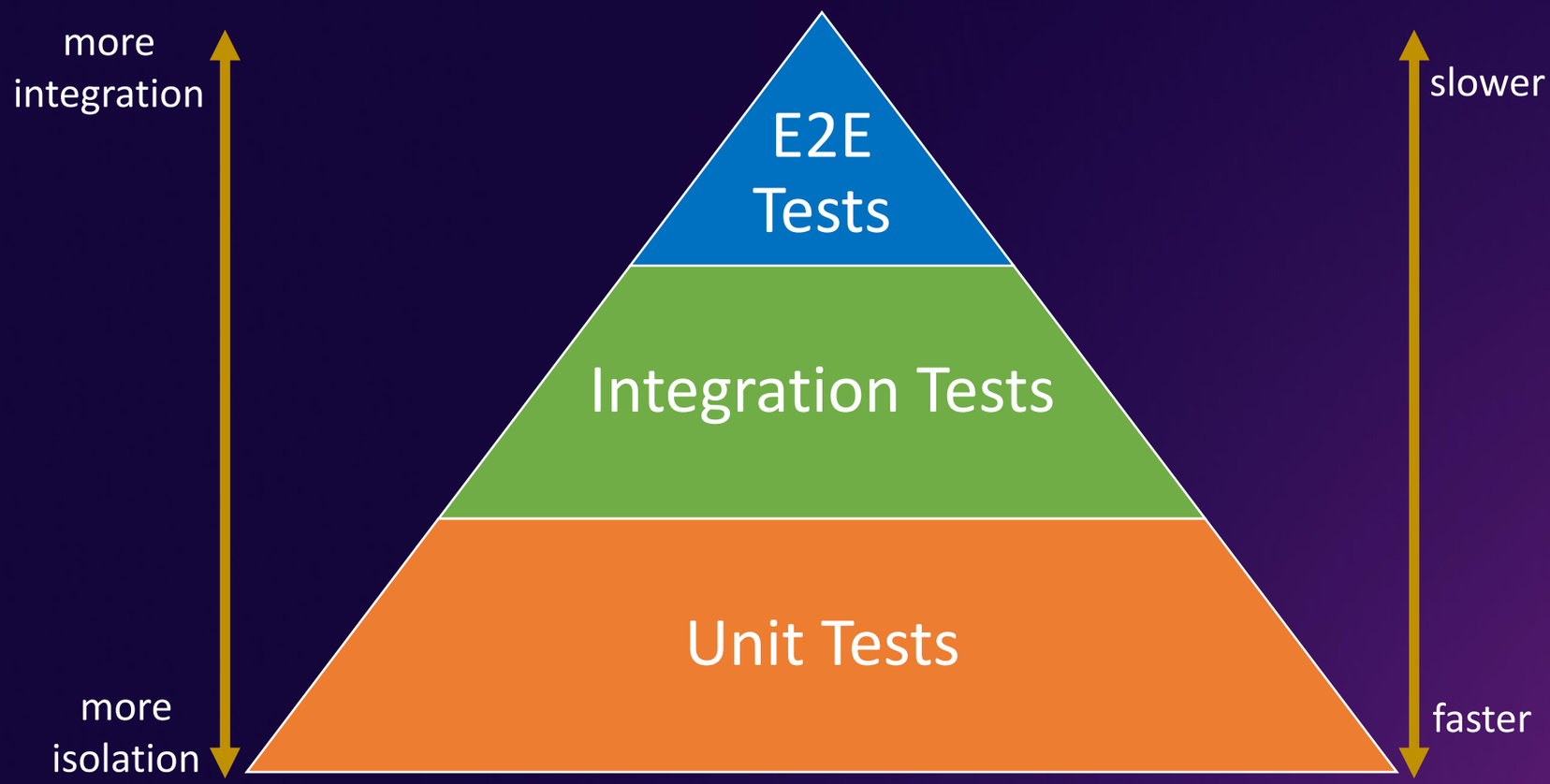
- Cheapest since don’t require any environment setup
- Quickest feedback cycle, usually in ms and on dev box
- The most reliable since are isolated from any external dependencies
- Can spot the exact line of the issue, reproduce locally and test the fix quickly
- Easy to maintain since are short and focused on a single unit

How many automated tests should I have?

The test pyramid



The test pyramid



Choosing a unit testing framework

What is a unit testing framework?

A software tool for creating, running and reporting unit tests.

Unit testing frameworks for C#/.NET

xUnit.net

MSTest

nunit

Why xUnit?

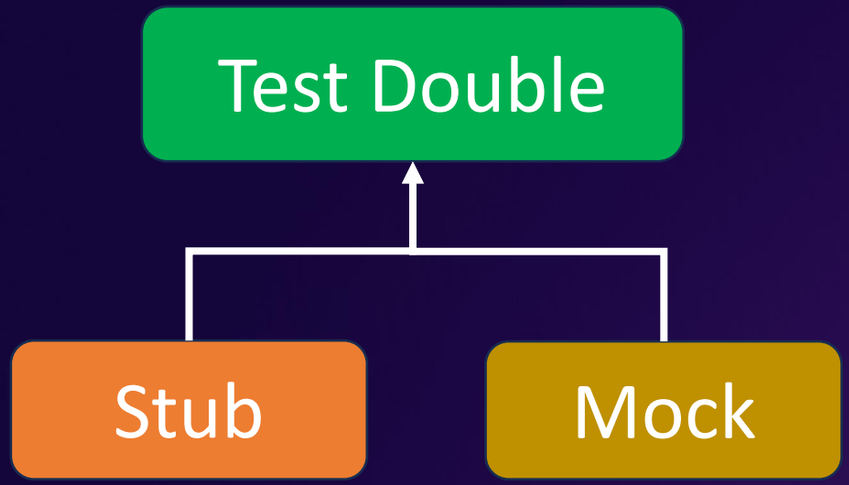
- ✓ Cleaner and more intuitive tests
- ✓ Runs tests in parallel by default
- ✓ Used by multiple Microsoft teams (ASP.NET Core, Entity Framework and others)

Stubs vs Mocks

What is a test double?

A placeholder or substitute for a real system component used in testing to isolate the unit of work.

Different types of test doubles



Stub

- Provides canned answers to calls made to dependencies during the test.
- Usually configured in the arrange section to return preconfigured responses

Mock

- Similar to a stub but also verifies the interactions with its dependencies.
- Usually used in the assert section to verify outgoing calls to dependencies